UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/539,624	03/31/2000	Jerrie L. Coffman	219.38025X00	9576

7590 11/10/2003

Rob D. Anderson
C/O Blakely, Sokoloff, Taylor, & Zafman LLP
12400 Wilson Boulevard
Seventh Floor
Los Angeles, CA 90025

EXAMINER	
ALI, SYED J	

ART UNIT	PAPER NUMBER
2127	8

DATE MAILED: 11/10/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

7

Office Action Summary	Application No.	Applicant(s)	
	09/539,624	COFFMAN ET AL.	
	Examin r	Art Unit	
	Syed J Ali	2127	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 02 September 2003 .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-21 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-21 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .	6) ☐ Other: .

DETAILED ACTION

1. This office action is in response to Amendment A, paper number 7, which was received September 2, 2003. Applicant's arguments have been fully considered but are moot in view of the new grounds of rejection. Claims 1-21 are presented for examination.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.\

3. Claim 1 is rejected under 35 U.S.C. 102(b) as being anticipated by Lindholm et al. (USPN 5,797,004) (hereinafter Lindholm).

As per claim 1, Lindholm discloses a system comprising:

a shared resource (col. 1 lines 30-45, "an object may require access to a shared computer resource, such as an I/O device, than can only handle one access at a time");

multiple processors arranged to access said shared resource (col. 1 lines 16-29, "In multiprocessor computer systems, software programs may be executed by threads that are run in parallel on the processors that form the multiprocessor computer system"); and

an operating system configured to allow said multiple processors to perform work on said shared resource concurrently while supporting state changes or updates of said shared resources (col. 3 line 54 - col. 4 line 2, "such methods may be explicitly declared synchronized...so that when one is executed by a thread, the object that contains the method will be synchronized with and owned by that thread", wherein the state changes for a shared resource correspond to the changing between a synchronized and non-synchronized state), said operating system comprising a synchronization algorithm for synchronizing multiple threads of operation with a single thread so as to achieve mutual exclusion between multiple threads performing work on said shared resource and a single thread updating or changing the state of said shared resource without requiring serialization of all threads (Fig. 3, wherein the method of a synchronization module synchronizing an object with only one thread is depicted) such that an update or change of the state of the shared resource may be made by the single thread only when none of the multiple threads are processing work on the shared resource (col. 11 lines 37-47, "a mutex is locked when it is allocated and has its synchronizers list 210 [or synchronizer identifier in the case where only one thread can by synchronized with an object] updated to identify the thread that is synchronized with an object and unlocked when it is de-allocated").

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 2-3 and 9-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lindholm in view of Spix et al. (USPN 5,179,702) (hereinafter Spix).

As per claim 2, Spix discloses the following limitations not shown by Lindholm, specifically the system as claimed in claim 1, wherein said shared resource includes work queues associated with a hardware adapter configured to send and receive message data to/from a remote system (col. 15 lines 3-27, "all of the CPU's...and all of the I/O controllers...have access to a common set of data structures in the Operating System Shared Resources [OSSR], including a Work Request Queue").

It would have been obvious to one of ordinary skill in the art to combine Lindholm with Spix since the method disclosed in Spix, while disclosing a way of allowing multiple CPUs to access shared resources that include work queues, does so according to an 'anarchistic' scheduling algorithm, which may be unsuitable for most needs. Thus, the combination with Lindholm provides a rigid scheduling algorithm that regulates access to synchronization constructs, while utilizing shared resources of particular I/O devices that support work queues, such as those associated with network communication.

As per claim 3, Lindholm discloses the system as claimed in claim 2, wherein said synchronization algorithm is executed to synchronize any thread wishing to update or change a state of said shared resource with all the threads processing I/O operations on said shared resource (col. 6 line 56 - col. 7 line 6, "If the cache manager finally encounters the last synchronization construct in the chain and its pointer does not point to another synchronization construct, as might be the case with synchronization constructs 124-3, then this means that the object 114-2 is not synchronized with any of the threads 116. In this case, the cache manager allocates the first synchronization construct 124-N in the free list", wherein when a thread attempts to lock a resource, the thread is synchronized with all threads in the system that may also be trying to lock that resource).

As per claim 9, Lindholm discloses the system as claimed in claim 2, wherein said synchronization algorithm is installed as part of a software driver module of an operation system [OS] kernel or an user-level application of said system (col. 3 lines 31-33, "the method employed by the object synchronization module...", wherein the synchronization algorithm is implemented within a software module).

As per claim 10, Spix discloses the system as claimed in claim 2, wherein said shared resource includes one of work queues, completion queues, FIFO queues, hardware adapters, I/O controllers and other memory elements of said system (col. 15 lines 3-27, "all of the CPU's...and all of the I/O controllers...have access to a common set of data structures in the Operating System Shared Resources [OSSR], including a Work Request Queue", wherein a FIFO queue is analogous to a work queue since a work queue could be designed as a FIFO as well as a LIFO, or any other queue structure).

Concerning the additional limitations of hardware adapters, I/O controllers, etc. Lindholm discloses these as types of shared resources that may require synchronization (col. 1 lines 30-45, "an object may require access to a shared computer resource, such as an I/O device, that can only handle one access at a time. Thus, since concurrently running threads may concurrently seek to invoke a method of an object, the object must be synchronized with only one thread at a time").

It would have been obvious to one of ordinary skill in the art to combine Lindholm with Spix for reasons discussed above in reference to claim 2.

6. Claims 4-7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lindholm in view of Kerrigan et al. (USPN 5,404,488) (hereinafter Kerrigan).

As per claim 4, Kerrigan discloses the following limitations not shown by Lindholm, specifically the system as claimed in claim 1, wherein said synchronization algorithm is executed to allow worker threads to work concurrently while processing I/O operations in exclusion of an update thread when a state of said shared resource is not changing, and allow an update thread to change the state or update said shared resource in exclusion of multiple worker threads (col. 24 lines 28-39, "the update thread sits on buffer access semaphore 3820 of the current buffer 3810 and waits for it to be cleared. When that semaphore is cleared, the underlying buffer area is released to the updater thread so that its messages may be transferred to the master interest list").

It would have been obvious to one of ordinary skill in the art to combine Lindholm with Kerrigan since Lindholm allows all threads that are synchronized with an object to perform state updates on the shared resource. This can lead to excessive context switching and tremendous overhead, a situation that requires remedy. Kerrigan discloses a way of updating an application using an update thread and a synchronization construct, specifically a semaphore. Although Kerrigan is related to updating the data pertaining to an application, the idea is easily combinable with Lindholm since Kerrigan discloses threads and synchronization constructs, as does Lindholm. The combination therein would thus allow Lindholm to be satisfactorily modified such that only one specific thread performs updates on the shared resource, thereby reducing the costly overhead incurred if each thread working on a shared resource updated the status upon synchronization or desynchronization.

As per claim 5, Lindholm discloses the system as claimed in claim 4, wherein said synchronization algorithm is executed to support a worker thread operation for processing simultaneous I/O operations on said shared resource while concurrently supporting an update thread operation for updating or changing the state of said shared resource (col. 1 lines 30-45, "an object may require access to a shared computer resource, such as an I/O device, that can only handle one access at a time. Thus, since concurrently running threads may concurrently seek to invoke a method of an object, the object must be synchronized with only one thread at a time", wherein the combination with Kerrigan as discussed in claim 5 provides for an update thread, and the synchronization algorithm would thus ensure that worker threads and the update thread are synchronized).

As per claim 6, Lindholm discloses a system as claimed in claim 5, wherein said worked thread operation is invoked by one of an event and a user's request, and is performed by:

determining whether a lock is available (col. 5 lines 7-20, "Since the thread 116-1 is seeking to synchronize the object 114-2...the cache manager determines if it points to any of the synchronization constructs");

if the lock is not available, waiting until the lock becomes available (col. 6 lines 22-34, "each synchronization construct 124 includes a waiters list 214 which identifies those of the threads 116 that are waiting to synchronize an object that is already synchronized by the synchronization construct with the predefined number of threads that can be synchronized with the object");

if the lock is available, seizing the lock while incrementing a count by a discrete constant to indicate the number of worker threads that are active, and then releasing the lock after the count has been incremented (col. 11 line 65 - col. 12 line 3, "In the case where the synchronization constructs 124 are semiphores, the predefined number of threads 116 that may be synchronized with one of the objects 114 at one time may be N", wherein as each thread is allocated to the synchronization construct the counter is incremented, so long as the value remains below N);

after the lock has been released, allowing multiple worker threads to process work concurrently (col. 3 line 55 - col. 7 line 20, wherein once the threads have been allocated to the synchronization construct, they concurrently execute on the resource);

determining next whether there is work to be processed (col. 7 lines 23-29, "When the synchronization method 200 of one of the objects 114, such as object 114-2, is terminated, synchronization of the object with whichever of the threads 116, such as 116-1, is completed");

if there is work to be processed, processing the work until there is no work to be processed (col. 7 lines 23-29, "When the synchronization method 200 of one of the objects 114, such as object 114-2, is terminated, synchronization of the object with whichever of the threads

116, such as 116-1, is completed", wherein execution continues until the method has completed); and

if there is no work to be processed, decrementing the count by a discrete constant to indicate when all the worker threads are done with completion processing (col. 7 line 47 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct allocated to the object", wherein as each thread releases the resources the synchronizers list is updated, until a determination is made that there are no more threads working on the resource).

As per claim 7, Lindholm discloses a system as claimed in claim 6, wherein said update thread operation is invoked by a user's request, and is performed by:

determining whether a lock is available (col. 5 lines 7-20, "Since the thread 116-1 is seeking to synchronize the object 114-2...the cache manager determines if it points to any of the synchronization constructs");

if the lock is not available, waiting until the lock becomes available when released by any one of the worker threads (col. 6 lines 22-34, "each synchronization construct 124 includes a waiters list 214 which identifies those of the threads 116 that are waiting to synchronize an object that is already synchronized by the synchronization construct with the predefined number of threads that can be synchronized with the object");

if the lock is available, seizing the lock until the count becomes zero (0) to indicate that it is safe to update or change the state of said shared resource, and updating or changing the state of said shared resource (col. 7 line 63 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct...by updating the pointer of the deallocated synchronization construct...and updating the free list header"); and

after said shared resource has been updated, releasing the lock so as to allow either new worker threads to continue I/O operation processing or a different update thread to continue shared resource updating (col. 7 line 63 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct...by updating the pointer of the deallocated synchronization construct...and updating the free list header", wherein updating the free list header tells the synchronization module that other threads may use the synchronization construct).

7. Claims 8, 11-12, and 17-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lindholm in view of Spix in view of Tillier (previously cited).

As per claim 8, Tillier discloses the following limitations not shown by the modified Lindholm, specifically a system as claimed in claim 2, further comprising data channels formed between said system and said remote system, via a switched fabric, and supported by the "*Virtual Interface [VI] Architecture Specification*" and the "*Next Generation Input/Output [NGIO] Specification*" for message data transfers between said system and said remote system (col. 5 lines 14-36, "The example embodiment and other embodiments of the invention can be implemented in conjunction with other types of switch fabric-based I/O architectures. The example embodiment NGIO uses a similar model for input/output data transfer as is specified by the VI architecture").

It would have been obvious to one of ordinary skill in the art to combine the modified Lindholm with Tillier since the modified Lindholm only speaks to synchronization of threads in a multiprocessor system, but does not necessarily account for other types of networks, such as a switched fabric network. Tillier discloses a way of implementing such systems and thus would allow the modified Lindholm to be implemented on a wider variety of systems.

As per claim 11, Lindholm discloses an operating system configured to allow said multiple processors to perform work on said shared resource concurrently while supporting state changes said shared resources (col. 3 line 54 - col. 4 line 2, "such methods may be explicitly declared synchronized...so that when one is executed by a thread, the object that contains the method will be synchronized with and owned by that thread", wherein the state changes for a shared resource correspond to the changing between a synchronized and non-synchronized state), said operating system comprising a synchronization algorithm for synchronizing multiple threads of operation with a single thread so as to achieve mutual exclusion between multiple threads performing work on said shared resource and a single thread changing the state of said shared resource without requiring serialization of all threads (Fig. 3, wherein the method of a synchronization module synchronizing an object with only one thread is depicted) such that an update or change of the state of the shared resource may be made by the single thread only when none of the multiple threads are processing work on the shared resource (col. 11 lines 37-47, "a mutex is locked when it is allocated and has its synchronizers list 210 [or synchronizer identifier in the case where only one thread can by synchronized with an object] updated to identify the thread that is synchronized with an object and unlocked when it is de-allocated").

Spix discloses the following limitations not shown by Lindholm, specifically that the shared resources may be work queues (col. 15 lines 3-27, "all of the CPU's...and all of the I/O controllers...have access to a common set of data structures in the Operating System Shared Resources [OSSR], including a Work Request Queue").

It would have been obvious to one of ordinary skill in the art to combine Lindholm with Spix since the method disclosed in Spix, while disclosing a way of allowing multiple CPUs to access shared resources that include work queues, does so according to an 'anarchistic' scheduling algorithm, which may be unsuitable for most needs. Thus, the combination with Lindholm provides a rigid scheduling algorithm that regulates access to synchronization constructs, while utilizing shared resources of particular I/O devices that support work queues, such as those associated with network communication.

Tillier discloses the following limitations not shown by the modified Lindholm, specifically a network, comprising:

a switched fabric (col. 5 lines 14-36, "The example embodiment and other embodiments of the invention can be implemented in conjunction with other types of switch fabric-based I/O architectures");

remote systems attached to said switched fabric (col. 2 lines 20-40, "The present invention is directed to emulation in an I/O unit when transferring data over a network" wherein it is inherent that a network would comprise remote systems); and

a host system comprising multiple processors; a host-fabric adapter provided to interface with said switched fabric and included work queues each configured to send and receive message data from a single remote system, via said switched fabric (fig. 1 element 101 – I_2O Host).

It would have been obvious to one of ordinary skill in the art to combine the modified Lindholm with Tiller since the modified Lindholm only speaks to synchronization of threads in a multiprocessor system, but does not necessarily account for other types of networks, such as a switched fabric network. Tillier discloses a way of implementing such systems and thus would allow the synchronization algorithm disclosed by Lindholm to be implemented on a wider variety of systems.

As per claim 12, Lindholm discloses the network as claimed in claim 11, wherein said synchronization algorithm is executed to synchronize any thread wishing to update or change a state or said work queues with all the threads processing I/O operations on said work queues (col. 6 line 56 - col. 7 line 6, "If the cache manager finally encounters the last synchronization construct in the chain and its pointer does not point to another synchronization construct, as might be the case with synchronization constructs 124-3, then this means that the object 114-2 is not synchronized with any of the threads 116. In this case, the cache manager allocates the first synchronization construct 124-N in the free list", wherein when a thread attempts to lock a resource, the thread is synchronized with all threads in the system that may also be trying to lock that resource).

As per claim 17, Tillier discloses the following limitations not shown by the modified Lindholm, specifically a network as claimed in claim 11, further comprising data channels formed between said system and said remote system, via said switched fabric, and supported by the "*Virtual Interface [VI] Architecture Specification*" and the "*Next Generation Input/Output*

[NGIO] Specification" for message data transfers between said host system and said remote systems (col. 5 lines 14-36, "The example embodiment and other embodiments of the invention can be implemented in conjunction with other types of switch fabric-based I/O architectures. The example embodiment NGIO uses a similar model for input/output data transfer as is specified by the VI architecture").

It would have been obvious to one of ordinary skill in the art to combine the modified Lindholm with Tillier for reasons discussed above in reference to claim 11.

As per claim 18, Lindholm discloses the network as claimed in claim 11, wherein said synchronization algorithm is installed as part of a software driver module of an operating system [OS] kernel or an user-level application of said host system (col. 3 lines 31-33, "the method employed by the object synchronization module...", wherein the synchronization algorithm is implemented within a software module).

As per claim 19, Tillier discloses the network as claimed in claim 11, wherein said host system and said remote systems represent channel endpoints of a data network implemented in compliance with the "*Next Generation Input/Output [NGIO] Specification*", and data channels formed between said host system and said remote systems, via said switched fabric, are supported by the "*Virtual Interface [VI] Architecture Specification*" and the "*Next Generation Input/Output [NGIO] Specification*" for message data transfers between said host system and said remote systems (col. 5 lines 14-36, "The example embodiment and other embodiments of the invention can be implemented in conjunction with other types of switch fabric-based I/O architectures. The example embodiment NGIO uses a similar model for input/output data transfer as is specified by the VI architecture", wherein the use of NGIO and VI architectures are meant example embodiments of switched fabric architectures that can be used, and support I/O messaging across networks).

It would have been obvious to one of ordinary skill in the art to combine the modified Lindholm with Tillier for reasons discussed above in reference to claim 11.

8. Claims 13-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lindholm in view of Spix in view of Tillier in view of Kerrigan.

As per claim 13, Kerrigan discloses the following limitations not shown by the modified Lindholm, specifically the network as claimed in claim 11, wherein said synchronization algorithm is executed to allow worker threads to work concurrently while processing I/O operations in exclusion of an update thread when the state of said work queues is not changing, and allow an update thread to change the state or update said work queues in exclusion of multiple worker threads (col. 24 lines 28-39, "the update thread sits on buffer access semaphore 3820 of the current buffer 3810 and waits for it to be cleared. When that semaphore is cleared, the underlying buffer area is released to the updater thread so that its messages may be transferred to the master interest list").

It would have been obvious to one of ordinary skill in the art to combine the modified Lindholm with Kerrigan since the modified Lindholm allows all threads that are synchronized with an object to perform state updates on the shared resource. This can lead to excessive context switching and tremendous overhead, a situation that requires remedy. Kerrigan discloses a way of updating an application using an update thread and a synchronization construct, specifically a semaphore. Although Kerrigan is related to updating the data pertaining to an application, the idea is easily combinable with Lindholm since Kerrigan discloses threads and synchronization constructs, as does Lindholm. The combination therein would thus allow Lindholm to be satisfactorily modified such that only one specific thread performs updates on the shared resource, thereby reducing the costly overhead incurred if each thread working on a shared resource updated the status upon synchronization or desynchronization.

As per claim 14, Lindholm discloses the network as claimed in claim 11, wherein said synchronization algorithm is executed to support a worker thread operation for processing simultaneous I/O operations on said work queues while concurrently supporting an update thread operation for updating or changing the state of said work queues (col. 1 lines 30-45, "an object may require access to a shared computer resource, such as an I/O device, that can only handle one access at a time. Thus, since concurrently running threads may concurrently seek to invoke a method of an object, the object must be synchronized with only one thread at a time", wherein the combination with Kerrigan as discussed in claim 5 provides for an update thread, and the synchronization algorithm would thus ensure that worker threads and the update thread are synchronized, and Spix provides that the shared resources may be work queues).

As per claim 15, Lindholm discloses a network as claimed in claim 14, wherein said worker thread operation is invoked by one of an event and a user's request, and is performed by:

determining whether a lock is available (col. 5 lines 7-20, "Since the thread 116-1 is seeking to synchronize the object 114-2...the cache manager determines if it points to any of the synchronization constructs");

if the lock is not available, waiting until the lock becomes available (col. 6 lines 22-34, "each synchronization construct 124 includes a waiters list 214 which identifies those of the threads 116 that are waiting to synchronize an object that is already synchronized by the synchronization construct with the predefined number of threads that can be synchronized with the object");

if the lock is available, seizing the lock while incrementing a count by a discrete constant to indicate the number of worker threads that are active, and then releasing the lock after the count has been incremented (col. 11 line 65 - col. 12 line 3, "In the case where the synchronization constructs 124 are semiphores, the predefined number of threads 116 that may be synchronized with one of the objects 114 at one time may be N", wherein as each thread is allocated to the synchronization construct the counter is incremented, so long as the value remains below N);

after the lock has been released, allowing multiple worker threads to process work concurrently (col. 3 line 55 - col. 7 line 20, wherein once the threads have been allocated to the synchronization construct, they concurrently execute on the resource);

determining next whether there is work to be processed (col. 7 lines 23-29, "When the synchronization method 200 of one of the objects 114, such as object 114-2, is terminated, synchronization of the object with whichever of the threads 116, such as 116-1, is completed");

if there is work to be processed, processing the work until there is no work to be processed (col. 7 lines 23-29, "When the synchronization method 200 of one of the objects 114, such as object 114-2, is terminated, synchronization of the object with whichever of the threads 116, such as 116-1, is completed", wherein execution continues until the method has completed); and

if there is no work to be processed, decrementing the count by a discrete constant to indicate when all the worker threads are done with completion processing (col. 7 line 47 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct allocated to the object", wherein as each thread releases the resources the synchronizers list is updated, until a determination is made that there are no more threads working on the resource).

As per claim 16, Lindholm discloses a network as claimed in claim 6, wherein said update thread operation is invoked by a user's request, and is performed by:

determining whether a lock is available (col. 5 lines 7-20, "Since the thread 116-1 is seeking to synchronize the object 114-2...the cache manager determines if it points to any of the synchronization constructs");

if the lock is not available, waiting until the lock becomes available when released by any one of the worker threads (col. 6 lines 22-34, "each synchronization construct 124 includes a waiters list 214 which identifies those of the threads 116 that are waiting to synchronize an object that is already synchronized by the synchronization construct with the predefined number of threads that can be synchronized with the object");

if the lock is available, seizing the lock until the count becomes zero (0) to indicate that it is safe to update or change the state of said shared resource, and updating or changing the state of said shared resource (col. 7 line 63 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct...by updating the pointer of the deallocated synchronization construct...and updating the free list header"); and

after said shared resource has been updated, releasing the lock so as to allow either new worker threads to continue I/O operation processing or a different update thread to continue shared resource updating (col. 7 line 63 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct...by updating the pointer of the deallocated synchronization construct...and updating the free list header", wherein updating the free list header tells the synchronization module that other threads may use the synchronization construct).

9. Claims 20-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lindholm in view of Spix in view of Kerrigan.

As per claim 20, Lindholm discloses a process of synchronizing an update thread which updates a list of shared resources with multiple worker threads which operate on items in the list of shared resources in a multiprocessor system, comprising:

allowing a group of worker threads to concurrently access the list of shared resources to process I/O operations in mutual exclusion, when states of the work queues are not changing

(Fig. 3, wherein the method of a synchronization module synchronizing an object with only one thread is depicted);

incrementing a count of threads processing I/O operations each time a worker thread is running (col. 11 line 65 - col. 12 line 3, "In the case where the synchronization constructs 124 are semiphores, the predefined number of threads 116 that may be synchronized with one of the objects 114 at one time may be N", wherein as each thread is allocated to the synchronization construct the counter is incremented, so long as the value remains below N), while decrementing the count of threads processing I/O operations each time a worker thread is done processing I/O operations (col. 7 line 47 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct allocated to the object", wherein as each thread releases the resources the synchronizers list is updated, until a determination is made that there are no more threads working on the resource);

when the count of threads reaches a designated value indicating that no worker threads are running, allowing an update to access and update the list of shared resources in exclusion of new worker threads from processing I/O operations (col. 7 line 63 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct...by updating the pointer of the deallocated synchronization construct...and updating the free list header"); and

after the list of shared resources is updated, allowing new worker threads to perform I/O operations until all worker threads are done processing I/O operations (col. 7 line 63 - col. 8 line 19, "if there are no more threads in the synchronizers list, the cache manager deallocates the synchronization construct...by updating the pointer of the deallocated synchronization construct...and updating the free list header", wherein updating the free list header tells the synchronization module that other threads may use the synchronization construct).

Spix discloses the following limitations not shown by Lindholm, specifically that the shared resources may be work queues (col. 15 lines 3-27, "all of the CPU's...and all of the I/O controllers...have access to a common set of data structures in the Operating System Shared Resources [OSSR], including a Work Request Queue").

It would have been obvious to one of ordinary skill in the art to combine Lindholm with Spix since the method disclosed in Spix, while disclosing a way of allowing multiple CPUs to access shared resources that include work queues, does so according to an 'anarchistic' scheduling algorithm, which may be unsuitable for most needs. Thus, the combination with Lindholm provides a rigid scheduling algorithm that regulates access to synchronization constructs, while utilizing shared resources of particular I/O devices that support work queues, such as those associated with network communication.

Kerrigan discloses the following limitations not shown by the modified Lindholm, specifically that updating the work queues should be done using an update thread (col. 24 lines 28-39, "the update thread sits on buffer access semaphore 3820 of the current buffer 3810 and waits for it to be cleared. When that semaphore is cleared, the underlying buffer area is released to the updater thread so that its messages may be transferred to the master interest list").

It would have been obvious to one of ordinary skill in the art to combine the modified Lindholm with Kerrigan since the modified Lindholm allows all threads that are synchronized with an object to perform state updates on the shared resource. This can lead to excessive context switching and tremendous overhead, a situation that requires remedy. Kerrigan discloses a way of updating an application using an update thread and a synchronization construct, specifically a semaphore. Although Kerrigan is related to updating the data pertaining to an application, the idea is easily combinable with Lindholm since Kerrigan discloses threads and synchronization constructs, as does Lindholm. The combination therein would thus allow Lindholm to be satisfactorily modified such that only one specific thread performs updates on the shared resource, thereby reducing the costly overhead incurred if each thread working on a shared resource updated the status upon synchronization or desynchronization.

As per claim 21, the modified Lindholm discloses a computer readable medium that stores computer executable instructions for implementing the process of claim 20. Specifically, since Lindholm discloses that the synchronization method should be implemented as a software module, the method inherently must reside on a computer readable medium that stores computer executable instructions in order to be functional.

Conclusion

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Syed J Ali whose telephone number is (703) 305-8106. The examiner can normally be reached on Mon-Fri 8-5:30, 2nd Friday off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, William A Grant can be reached on (703) 308-1108. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

[illegible]

Syed Ali
November 4, 2003

[illegible]

WILLIAM GRANT
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100
11/6/03